A Reference Model for Virtual Machine Launching Overhead

Hao Wu*, Shangping Ren*, Gabriele Garzoglio†, Steven Timm†, Gerard Bernabeu†,Keith Chadwick†,Seo-Young Noh‡

Abstract—*Cloud bursting* is one of the key research topics in the cloud computing communities. A well designed *cloud bursting* module enables private clouds to automatically launch virtual machines (VMs) to public clouds when more resources are needed. One of the main challenges in developing cloud bursting module is to decide *when* and *where* to launch a VM so that all resources are most effectively and efficiently utilized and the system performance is optimized. However, based on system operational data obtained from the FermiCloud, a private cloud developed by the Fermi National Accelerator Laboratory for scientific workflows, the VM launching overhead is not a constant. It varies with physical resource utilization, such as CPU and I/O device utilizations, at the time when a VM is launched. Hence, to make judicious decisions as to *when* and *where* a VM should be launched, a VM launching overhead reference model is needed. In this paper, we first develop a VM launching overhead reference model based on operational data we have obtained on the FermiCloud. Second, we apply the developed reference model on the FermiCloud and compare calculated VM launching overhead values based on the model with measured overhead values on the FermiCloud. Our empirical results on the FermiCloud indicate that the developed reference model is accurate. We believe, with the guidance of the developed reference model, efficient resource allocation algorithms can be developed for cloud bursting process to minimize the operational cost and resource waste.

Index Terms—VM Launching Overhead, Reference Model, Cloud, FermiCloud, Virtual Machine, VM Launching,VM Startup Time, Launch, Overhead, Model, Predict

1 Introduction

Cloud technology has been benefiting general purpose computing for a number of years. The *pay-on-demand* model brought about by cloud computing allows companies to avoid over-provisioning in early stages of project development. Furthermore, comparing to the traditional grid computing, cloud computing can better utilize resources provided by its underlying infrastructure, as it can deploy different tasks on the same physical computer node. In addition, computation power can also be dynamically allocated to tasks when more resources are needed by the tasks. Another benefit of using a cloud over a grid is that a cloud has "unlimited" resources – when a private cloud is fully occupied, cloud bursting techniques can temporarily acquire external resources from public clouds to fulfill the need.

Many scientific research institutions have foreseen the benefits of using computer clouds and have migrated their research platforms from traditional grid and distributed computing platform to the cloud computing environment [1] [12] [17] [19] [10]. Fermi National Accelerator Laboratory (Fermilab), a leading research institution in the high energy physics (HEP) field, started to build a private infrastructure-as-a-service facility, the FermiCloud, in 2010. The FermiCloud has successfully served the HEP experiments since its establishment. S. Y. Noh *et al.* [20] [16] of KISTI collaboratively developed the *vcluster* cloud management tool for FermiCloud to automatically allocate cloud cycles on FermiCloud and KISTI's GCloud as well as Amazon AWS. However, how to dynamically allocate resources so that application's average response time and system's total operational cost are reduced is a research and engineering challenge yet to be addressed.

Resource allocation problems in cloud computing has drawn more and more attention in research community in recent years [14], [7]. However, most research in the area assume that VM launching overheads with respect to time and resource consumption are negligible. As a result of this assumption, neither the launching overhead nor the dependency between the overhead and resource utilization are taken into consideration in designing resource allocation algorithms. However, our production line operation data (Fig. 1) indicates that the VM launching overhead can have significant variations when it is launched at different time or on different physical machines. Figure 1 depicts the launching time for 227 VMs that have been deployed on FermiCloud over one month period. The VM launching time ranges from few seconds to over one thousand seconds.

In addition to time overhead, VM launching overhead also includes system resources a VM consumes during its launching process. Both of time and utilization overheads can impact the system's performance. For instance, VM launching process consumes a significant amount of CPU and I/O resources, leads to a high system CPU and I/O

*Illinois Institute of Technology,10 W 31st street, 013, Chicago, IL, USA, {hwu28, ren}@iit.edu.
†Fermi National Accelerator Laboratory, Batavia, IL, USA. {garzogli,timm,gerard1,chadwick}@fnal.gov.
‡National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon, Korea, rsyoung@kisti.re.kr
The research is supported in part by NSF under grant number CAREER 0746643 and CNS 101873, by the U.S. Department of Energy under contract number DE-AC02-07CH11359 and by KISTI under a joint Cooperative Research and Development Agreement CRADA-FRA 2013-0001 / KISTI-C13013.

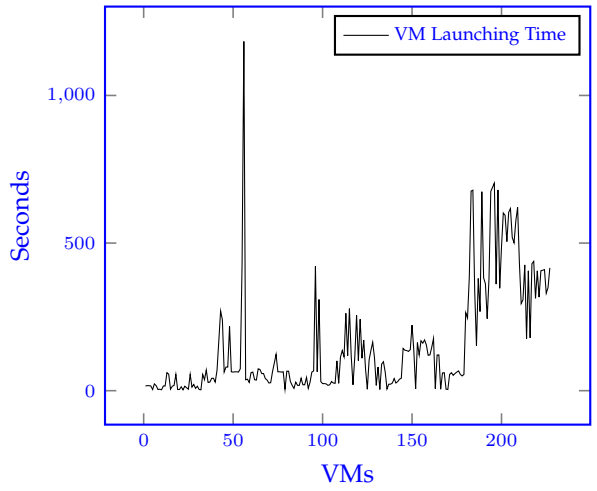


Fig. 1: VM Launching Time on FermiCloud (08/22/2014 - 09/19/2014)

utilization at the time of launching. If each host computer in the private cloud happens to launch a new VM at the same time, due to high system utilization caused by VM creation, the computer cloud may consider all its hosts are fully occupied and decide to perform cloud bursting and create VMs on an external public cloud. Such additional cost of bursting to external public cloud is unnecessarily rendered and can be prevented if we have a VM launching overhead reference model. Furthermore, if a task requires an additional VM in order to complete its work, but the VM takes much longer time to complete its launching than expected, it is possible that the task has already finished its work before the VM is ready for executing the task. This again leads to resource waste and an added cost due to the lack of a VM launching overhead information.

In this paper, we are to 1) analyze the patterns of VM launching process based on large amount of data obtained from real working systems, 2) define a reference model to represent the patterns, and 3) validate the accuracy of the developed reference model with real system operational data. At this initial study stage, we emphasize a methodological point of view rather than definitive numerical results based on accurate parameter values. The main purpose of this study is to show how an analytic model can be developed.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 introduces the FermiCloud and its architecture. Section 4 defines the terms that are used in the paper. Section 5 analyzes the VM launching overhead based on a large set of experiments on the FermiCloud. Section 6 presents a reference model for VM launching overhead. Section 7 evaluates the accuracy of the proposed model. We conclude the work in section 8.

2 Related Work

Researches on modeling and evaluating the cloud's performance started almost at the same time when computer cloud itself emerged. One of the most influential cloud modeling tool is CloudSim [7] developed by the CLOUDS lab from the University of Melbourne. CloudSim is a Java-based cloud simulation tool that supports modeling and simulation of large scale cloud computing environments. The CloudSim provides a comprehensive modeling tool that covers almost all basic elements under a cloud environment. In particular, it provides an infrastructure modeling to capture the characteristics and behaviors of both VM and physical infrastructure where VMs are deployed. It also provides a cloud market model that models the cost of resources, a network model that models the network behavior of inter-networking of clouds, a cloud federation model that models the communication between clouds, a power consumption model that models the power consumptions in the datacenter, and a resource allocation model that models VM allocation policies.

Recently, Huber *et al.* evaluate the virtualization performance and propose a virtualization overhead model [9]. In their work, they mainly focus on two virtualization platforms, i.e., XenServer and VMware ESX. They test the performance downgrades that are brought by the virtualization. They test the CPU, memory, disk IO, and network performance degradations on both XenServer and VMware ESX platforms. Based on the experiments, they categorize the virtualization performance influencing factors into four major categories: virtualization type, hypervisor's architecture, resource management configuration, and workload profile. However, Huber's model does not consider virtual machine launching overhead, it only provides the computation overhead caused by the virtualization.

Researchers have adapted the above cloud models and cloud simulations tools and made significant contributions to optimize resource allocation process on clouds, such as resources provisioning algorithms from QoS perspective [8], from service providers' profit perspective [18], and from energy consumption perspective [6]. Recently, Mengxia Zhu *et al.* have proposed a cost effective scheduling for scientific workflow under cloud environments [14]. Their scheduling algorithm aims to shorten the application's response time and reduce the energy consumption simultaneously by considering VM launching overhead. Some of the researches, such as Zhu's work [14] and CloudSim [7], have taken VM launching overhead variation as a key variable for designing resource allocation algorithms. However, they treat the VM launching overhead as a constant.

Some of the researchers have observed that VM overhead may have a large variation on public cloud and realized that the variation of VM launching overhead may cause significant impact on the resource allocation process [13] [11]. Hence, significant contributions have been made on reducing the impact of VM launching overhead. For instance, Lagar-Cavilla *et al.* [11] have developed a cloud programming paradigm and a system called SnowFlock that can significantly improve the VM scaling efficiency using fast VM cloning.

We have also observed significant VM launching overhead variations on the FermiCloud's daily operations. In the FermiCloud bursting project, the design of the resource allocation algorithm aims to automatically allocate resources for applications that need extra computational resources. If the VM launching overhead variation is not well modeled and calculated, the system utilization and efficiency may

be pulled down dramatically, causing resource and energy waste. Hence, we need an accurate mathematical model for VM launching overhead. Rather than aiming to reduce VM launching overhead, the goal of this paper is to understand and analyze VM launching process and developed a reference model to predict the overhead during such process.

3 FermiCloud

The FermiCloud uses OpenNebula [3], [15] as its cloud infrastructure management tool and uses KVM as its VM management tool. VMs running on the FermiCloud are all paravirtualized. Under OpenNebula [4], the VM launching process consists of four major states. Fig. 2 illustrates the state change during a VM launching process in OpenNebula [4]. In particular, when a user creates a new VM, the VM enters the *pending* state. In the pending state, the cloud scheduler decides where to deploy the VM. Once the VM is deployed on a specific host, it enters into the *prologue* state in which all VM related files (images in our case) are transferred from the image repository to the host machine. After all the files are copied to the host, the VM enters the *boot* state, during which it is booted from the host. Finally, after the VM is successfully booted, it enters into the *running* state. Once a VM is in its running state, it is ready to execute tasks.

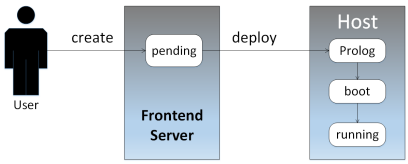


Fig. 2: VM Launching State Diagram[4]

Fig. 3 illustrates the system architecture of FermiCloud. The FermiCloud system has an OpenNebula front-end server that manages the entire cloud infrastructure, an image repository that stores the VM images, and a set of host machines on which VMs are deployed. Both front-end server and VM hosts use the GFS2 clustered shared file system, which is hosted on a fibre-channel connecting SAN with two NexSan SataBeast servers for storage servers. The logical unit used in the study has ten 7200-RPM SATA disk drives in a 9+1 RAID5 configuration for 17TB of usable space. Each SAN controller has 2GB cache memory. Both front-end server and VM hosts are configured with 16-core Intel(R) Xeon(R) E5640 @ 2.67GHz CPU and 48GB memory. All the machines in FermiCloud are installed with Scientific Linux operating system [2].

4 Terminology

This section defines the terms used in the paper.

Host CPU utilization: The host CPU utilization is defined as total CPU utilizations consumed by all the processes on one host machine.

Host disk write utilization: The host disk write utilization is defined as the total disk write bandwidth utilizations consumed by all the processes on one host machine.

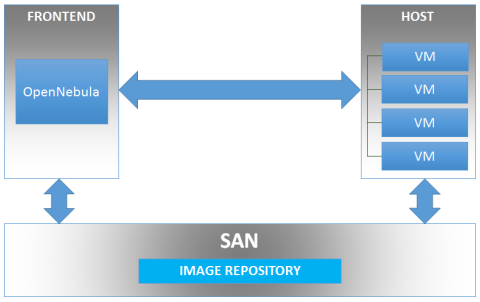


Fig. 3: System Architecture

Host disk read utilization: The host disk read utilization is defined as the total disk read bandwidth utilizations consumed by all the processes on one host machine.

VM CPU utilization: The VM CPU utilization is defined as the CPU utilization consumed by a VM on a single core. For example, in a 16-core CPU machine, the VM CPU utilization is 100% means the VM fully occupies one core. It equals to the consumption of 6.25% host CPU utilization.

VM disk write utilization: The VM disk write utilization is defined as the amount of disk write bandwidth consumed by a VM over the total disk write bandwidth.

VM disk read utilization: The VM disk read utilization is defined as the amount of disk read bandwidth consumed by the VM over the total disk read bandwidth.

Prologue: Prologue is an OpenNebula [4] terminology that indicates the process of copying an image from image repository to host machine. In the paper, the term *prologue* is interchangeable with the term image transmission process.

5 VM Launching Overhead on FermiCloud

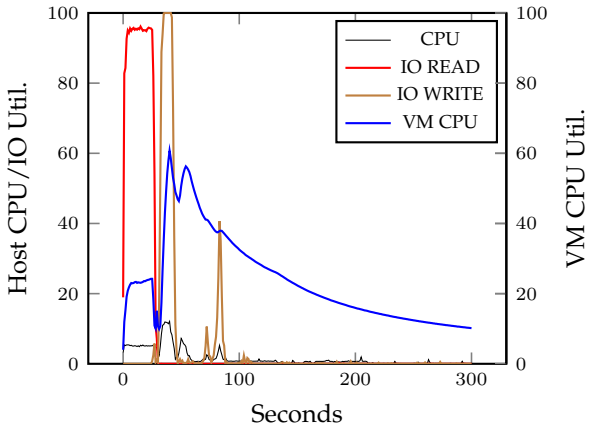
In this section, we study the patterns of VM launching overhead based on the VM operations in the FermiCloud environment.

5.1 VM Preparation

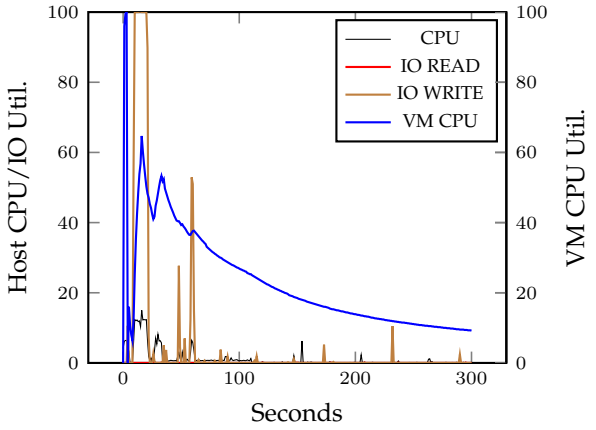
In our experiment tests, we focus on two types of VM instances, i.e., a small instance configured with one virtual CPU core and 2GB memory, and a large instance configured with 16 virtual CPU cores and 32GB memory. There are also two types of VM images tested, i.e., 4.7GB "QEMU Copy On Write 2" (QCOW2) image and 15.6GB raw image. Hence, four different types of VMs are tested during the experiments, i.e., small instance VM with QCOW2 image (SQ), small instance VM with raw image (SR), large instance VM with QCOW2 image (LQ) and large instance VM with raw image (RL).

5.2 Methodology

We use *noninvasive* programs, i.e., *iostat* and *sar* to obtain system information. The OpenNebula platform logs the time points when VMs enter the *running* state. In order to minimize the impact of cloud management tools, i.e., OpenNebula in our case, on VM launching process, we count VM start time as the time when a VM is deployed on the host machine. In cloud environment, VMs are not considered to be ready for use until the users can access the

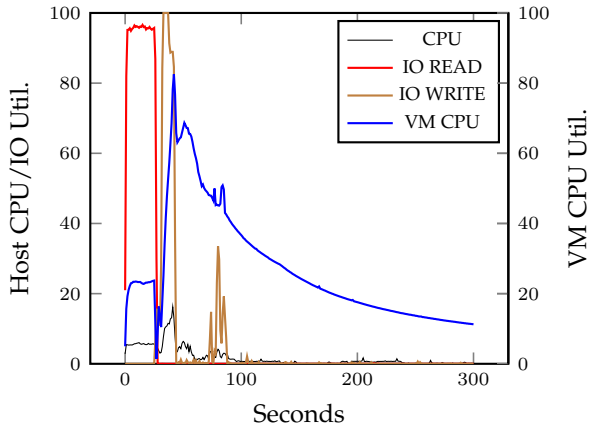


(a) Utilization Variation of Small Instance with Uncached QCOW2 Images

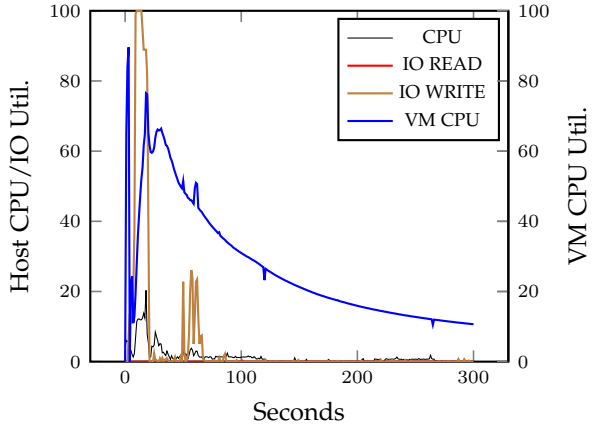


(b) Utilization Variation of Small Instance with Cached QCOW2 Images

Fig. 4: Utilization Variation of Small Instance with QCOW2 Images



(a) Utilization Variation of Large Instance with Uncached QCOW2 Images



(b) Utilization Variation of Large Instance with Cached QCOW2 Images

Fig. 5: Utilization variation of Large Instance with QCOW2 Images

VMs. Hence, we retrieve the start time of SSHD service from VMs' system logs as the times when VMs are actually ready for use.

In the cloud environment, existing running VMs may have significant impact on VM launching overhead. However, from physical machine (VM host machine)'s point of viewer, all VMs that are running on it are processes. Applications that are running on different VMs are transparent to physical machines. Hence, resources consumed by the VMs and workloads inside the VMs are reflected by the consumption of physical resources. Hence, we mimic the scenario that VM is launched when there are VMs already running in system by manipulating host machine's physical resource usage, i.e. CPU and I/O utilizations.

5.3 Base VM Launching Overhead

We first obtain the baseline overhead of launching a new VM. In order to obtain the baseline utilization overhead of launching a new VM, we let all the host machines in the private cloud be empty, i.e., have no application being deployed on the cloud. Each time, a single VM is launched on an empty host machine. The experiment is repeated twenty times for each type of VM.

Fig. 4(a), Fig 4(b), Fig. 5(a), Fig 5(b), Fig. 6(a), Fig. 6(b), Fig. 7(a) and Fig. 7(b) illustrate the average utilization changes for SQ, LQ, SR and RL VMs, respectively. The blue line depicts the VM CPU utilization, the black line the host CPU utilization, the brown line the host disk write utilization, and the red line the host disk read utilization. Table 1 gives the statistics of the utilization and timing variations of baseline VM launching processes.

5.3.1 *Cached and Uncached Images*

File cache is a Linux operating system feature that usually happens in file copying process. The VM launching process first copies VM's image from an image repository to the host machine. Once the VM image is copied onto the host machine, it is also cached into host machine's memory. If the VM being launched on the host machine has the same image as the cached one, the VM's image is directly copied from the memory instead of copied from an image repository. Hence, for the four types of tested VMs, we also test the launching overheads when launch them from both cached images and uncached images. For convenience, we use SQ_U, SQ_C, LQ_U, LQ_C, SR_U, SR_C, LR_U and LR_C to denote the test cases of small instance VM with

TABLE 1: Statistics of Base VM Launching Process

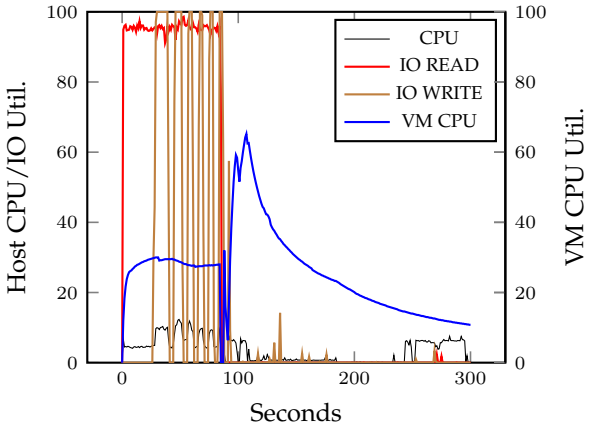
	SQ_U	SQ_C	LQ_U	LQ_C	SR_U	SR_C	LR_U	LR_C
LAUNCH TIME (Sec.)	74.10	49.40	73.56	51.30	129.80	70.80	135.44	71.50
MAX	79.00	53.00	75.00	56.00	136.00	77.00	140.00	74.00
MIN	72.00	46.00	71.00	47.00	126.00	67.00	132.00	68.00
TRANS. TIME (Sec.)	28.80	4.80	28.30	4.30	87.00	19.80	89.30	19.70
MAX	31.00	5.00	30.00	5.00	90.00	21.00	91.00	21.00
MIN	28.00	4.00	27.00	4.00	88.00	19.00	88.00	19.00
BOOT TIME (Sec.)	45.30	44.60	45.33	47.00	42.80	51.00	46.11	51.80
MAX	48.00	48.00	47.00	51.00	52.00	57.00	52.00	54.00
MIN	44.00	41.00	43.00	42.00	38.00	47.00	44.00	49.00
PEAK Util. (%)	62.83	64.56	81.20	83.42	58.83	57.76	78.28	75.97
MAX	65.40	66.50	85.00	90.70	65.90	59.80	81.90	78.80
MIN	54.30	62.90	78.40	80.60	53.40	55.60	70.90	72.10
TRANS. Util. (%)	24.55	75.89	24.97	72.72	25.91	78.91	25.00	78.40
MAX	26.11	98.50	27.36	79.53	27.56	81.98	28.13	81.25
MIN	22.13	66.13	22.68	66.25	23.77	74.78	23.53	73.96

uncached QCOW2 image, small instance VM with cached QCOW2 image, large instance VM with uncached QCOW2 image, large instance VM with cached QCOW2 image, small instance VM with uncached raw image, small instance VM with cached raw image, large instance VM with uncached raw image and large instance VM with cached raw image, respectively. Fig. 4(a), Fig. 5(a), Fig. 6(a) and Fig. 7(a) illustrates the utilizations changes for SQ_U,LQ_U, SR_U, and LR_U, respectively. While Fig. 4(b), Fig. 5(b), Fig. 6(b) and Fig. 7(b) illustrates the utilizations changes for SQ_C, LQ_C, SR_C and LR_C, respectively.

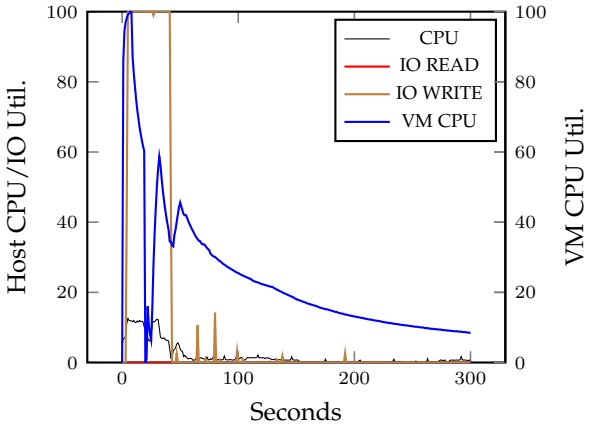
5.3.2 VM Prologue Process

The VM prologue process that copies the VM image from the image repository to the host machine is the first step of the entire VM launching process. As the measured disk read bandwidth of SAN in FermiCloud is 180MB/s, the measured disk write bandwidth of SAN in FermiCloud is 400MB/s. The theoretical time of transferring a 4.7GB QCOW2 image and a 15.6GB raw image from the image repository to the host machine is 26 seconds and 87 seconds, respectively. As indicated in Table 1, the average prologue time of SQ_U and LQ_U are 28.8 seconds and 28.3 seconds, respectively; the average prologue time of SR_U and LR_U are 87 seconds and 89.3 seconds, respectively. The measured prologue times align well with the calculated values. It is also clear from Fig. 4(a), 5(a), 6(a) and 7(a) that during the prologue time, disk read utilization almost constantly reaches 100% of the SAN read bandwidth (as shown by the red lines in the figures). The measured local disk read bandwidth is 500MB/s. When the local disk read bandwidth is fully utilized, the host CPU utilization consumption is around 70% on single core. Hence, the calculated VM CPU utilization during the prologue process is $180/(500/0.7) = 25.2\%$. The calculated value matches the measured average VM CPU utilization, which is around 25%.

Notice from the figures that the disk write activities are not synchronized with the disk read activities. In Linux, when a file is copied, the file is written into dirty pages first, it is then flushed into the hard disk. By Linux default settings, the flushing process happens if the dirty page size reaches 10% of the system active memory or 30 seconds after the content is written into the dirty page. As the 4.7GB QCOW2 image is smaller than the size of 10% of system memory (48GB * 10%) and the transmission time of the image is less than 30 seconds. The flush process is activated immediate after the whole image is copied into the dirty page (as illustrated in Fig. 4(a) and Fig. 5(a)). For large images, i.e., 16GB raw images, as illustrated in Fig. 6(a) and Fig. 7(a), the flushing processes are activated

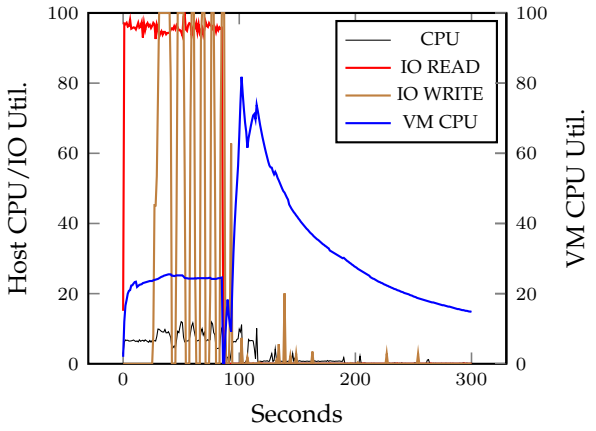


(a) Utilization Variation of Small Instance with Uncached Raw Images

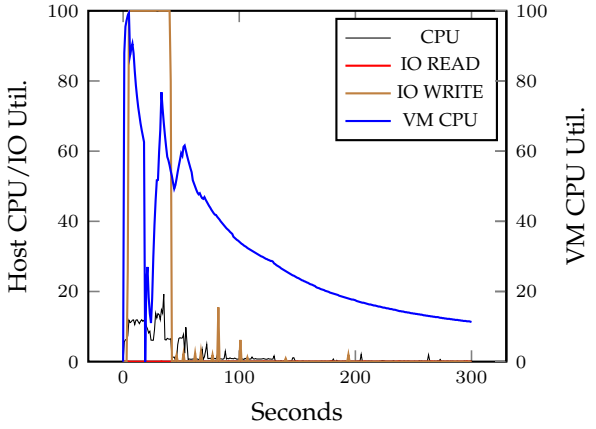


(b) Utilization Variation of Small Instance with Cached Raw Images

Fig. 6: Utilization Variation of Small Instance with Raw Images



(a) Utilization Variation of Large Instance with Uncached Raw Images



(b) Utilization Variation of Large Instance with Cached Raw Images

Fig. 7: Utilization Variation of Large Instance with Raw Images

after 30 seconds or when the dirty page reaches 10% limits (whenever happens first).

There is an interesting observation from Fig. 6(a) and Fig. 7(a) that the write process is not constantly writing images into the disk. This is because the write speed (400MB/s) is much faster than the read speed (180MB/s), by Linux default, the flushing process is awaken every 5 seconds if such scenario happens.

On the other hand, for cached images, as illustrated in Fig. 4(b), 5(b), 6(b) and 7(b), there are no disk read activities happening during the VM prologue processes. This is because images are read directly from the memory. As the measured cache read speed in FermiCloud host machines are around 1.2 GB/s. The theoretical transmission time for QCOW2 and raw images are 3.9 seconds and 18.5 seconds, respectively. As given in Table 1, the measured transmission times for cached images are consistent with the calculated values. Notice that, when images are read from memory, it takes about 4 seconds for the dirty pages reaches 10% limit. Hence, the flushing process happens after 4 seconds of the VM prologue process, and images are continuously being written into the disk. Since the prologue process fully utilize the memory bandwidth, the VM CPU utilization during the prologue process for cached images also reaches 100%.

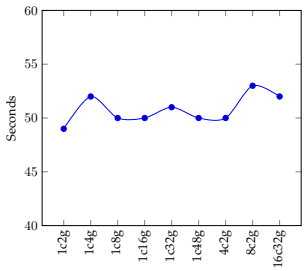


Fig. 8: VM Booting Time Comparison for Different Configurations

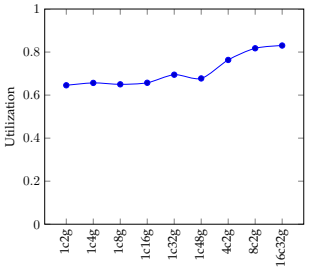


Fig. 9: VM Booting Peak CPU Utilization Comparison for Different Configurations

5.3.3 VM Boot Process

After the image is copied into the host machine, the VM then enters the boot process. Once the VM is booted, it can be used by end users. Note that, the boot process starts immediately after the image is copied into cache, not after the entire image is written into the disk. From the figures we can tell that the patterns of VM CPU utilization variation for all test cases are similar. The VM CPU utilization soon reaches a peak after the boot process begins, then VM CPU utilization slowly decreases and remains a low level. Table 1 also shows that the booting times for all cases are almost the same. The only difference among the different test cases is the peak VM CPU utilization. For the small instance VMs, the peak VM CPU utilization is around 60% of single core CPU utilization. However, for the large instance VMs, the peak VM CPU utilization is around 80% of single core CPU utilization.

5.3.4 VM Boot process comparison under different configurations

From the above experiments, we observe that different VM configurations do not affect the VM booting time. However, the configurations change the peak VM CPU utilization during the booting process. In order to understand how the configuration affects the peak VM CPU utilization during the booting process, we perform a more detailed test on varies VM configurations. We first test the VMs with 1 virtual core and change the memory from 2GB to 48 GB. Then we configure the VMs with 2GB memory and change the number of virtual cores from 2 to 16. All VMs are launched with QCOW2 images. The booting times of different VMs are depicted in Fig. 8. As can be seen from the figure, the booting times of VMs with different configurations vary only within a small range. We consider them to have the same booting times. Fig. 9 depicts the peak VM CPU utilization of VMs with different configurations during the booting process. It clearly shows that the memory configuration change does not affect the peak VM CPU utilization. However, the peak VM CPU utilization increases as the number of virtual cores increases.

5.4 CPU Utilization Impact

In this experiment, VMs are launched under different host CPU utilizations. We write a small bash script that con-

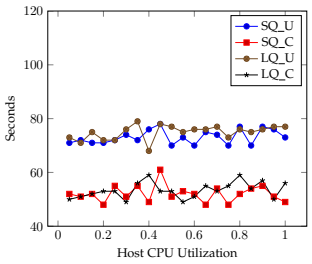


Fig. 10: VM Booting Time under Different CPU Utilization for QCOW2 Image

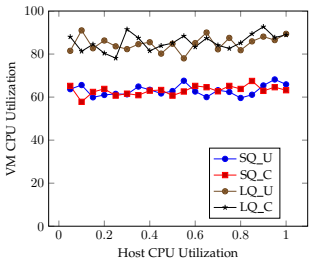


Fig. 11: Peak VM CPU Utilization under Different CPU Utilization for QCOW2 Image

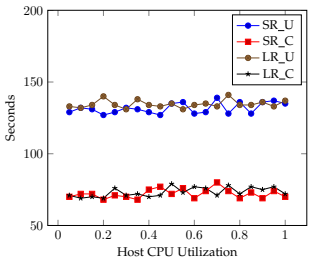


Fig. 12: VM Booting Time under Different CPU Utilization for Raw Image

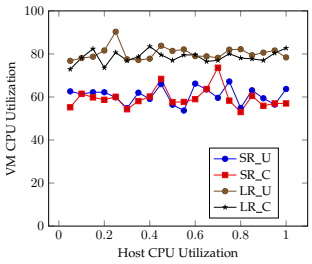


Fig. 13: Peak VM CPU Utilization under Different CPU Utilization for Raw Image

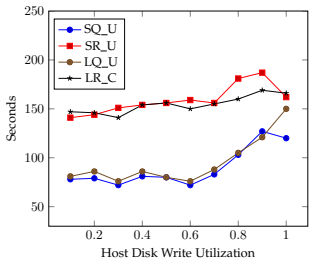


Fig. 14: VM Launching Time under Different IO Utilization using Uncached Images

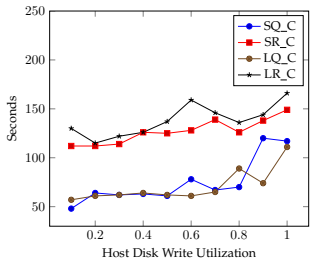


Fig. 15: VM Launching Time under Different IO Utilization using Cached Images

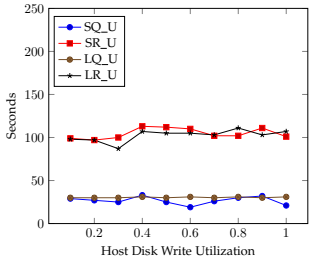


Fig. 16: VM Prologue Time under Different IO Utilization using Uncached Images

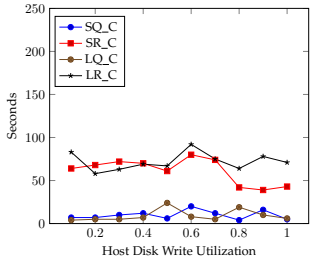


Fig. 17: VM Prologue Time under Different IO Utilization using Cached Images

stantly consume the host CPU utilization. We use *cgroup* to control the CPU usage of the script. Each time, we increase 5% of the CPU utilization consumed by the script. Hence, we can simulate the scenario that VMs are launched under different host CPU utilizations.

Fig. 10 depicts the launching times of VMs with QCOW2 images. As can be seen from the figure, the launching time of the VMs are relatively steady. For the uncached images, the maximum launching time is 79 seconds while the minimum launching time is 68 seconds for both small and large VM instances. For the cached images, the range of the VMs' launching times is from 48 seconds to 61 seconds. Comparing with the base VM launching times listed in Table 1, we consider the variations of the launch times are in normal condition. Hence, we can conclude that the host CPU utilization does not impact the VMs' launching times. As illustrated in Fig. 11, the peak VM CPU utilization during the booting process are also quite steady(60% for the small instances and 80% for the large instances). Hence, the host CPU utilization does not impact the peak VM CPU utilization during the booting process.

Fig. 12 and Fig. 13 illustrate the launching times and peak VM CPU utilization under different host CPU utilizations for the raw images. Similar to the QCOW2 images, the launching times and peak VM CPU utilizations are insensitive to the host CPU utilization changes. Hence, we can conclude that the host CPU utilization does not impact the VMs' launching process.

5.5 Disk Write Utilization Impact

In this experiment, we test the VM launching overhead under different host disk write utilizations. We use *dd* command to constantly write files to the disk., and use *cgroup* to control the write speed of the *dd* command. At each step, we increase 10% of the disk write bandwidth for *dd* command. Hence, we can simulated the scenario that VMs are launching under different host disk write utilizations.

Fig. 14 and Fig. 15 show the VMs' launching times under different host disk write utilizations. Overall, when the host disk write utilization increases, VMs need more time to be launched as compared with the base VM launching times. However, for the cached images, i.e., as illustrated in Fig. 15, the increasing trends are not so obvious when compared with the uncached images(Fig. 14). As for the VM prologue times, the prologue times for uncached images are steady and close to a constant. However, if the image is cached, the VMs' prologue times have larger variations. For the QCOW2 images, the variation is from 4 seconds to 24 seconds, and for a raw image, the variation is from 39 seconds to 92 seconds. This large variation is caused by cache override. Since the *dd* command also write files into the cache, it is possible that the cached image is overridden by the *dd* command. Hence, the missing part of the image need to be re-transferred from the data repository. In the worst case, the entire cached image is overridden by the *dd* command, and the entire image needs to be copied from the image repository.

Fig. 18 and Fig. 19 illustrate the booting times for un-

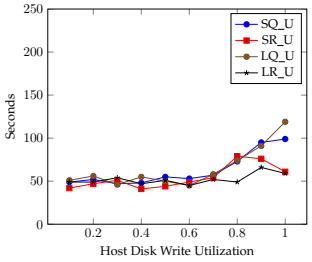


Fig. 18: VM Booting Time under Different IO Utilization using Uncached Images

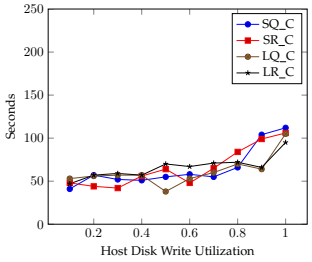


Fig. 19: VM Booting Time under Different IO Utilization using Cached Images

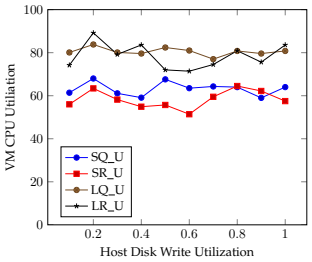


Fig. 20: VM Peak CPU Utilization under Different IO Utilization using Uncached Images

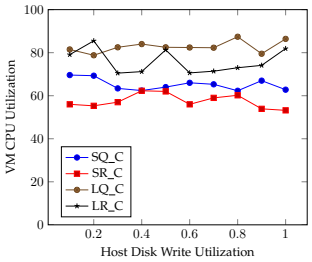


Fig. 21: VM Peak CPU Utilization under Different IO Utilization using Cached Images

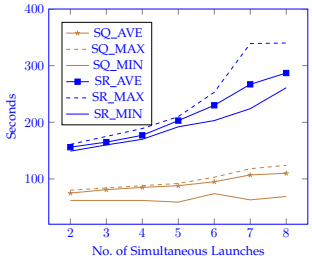


Fig. 22: VM Launching Time under Different Simultaneous Launches

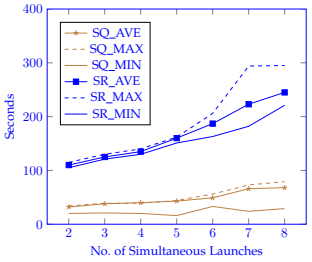


Fig. 23: VM Prologue Time under Different Simultaneous Launches

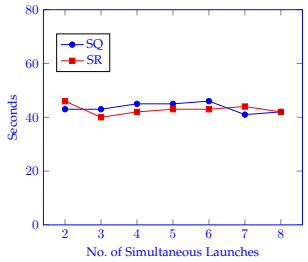


Fig. 24: Ave. VM Booting Time under Different Simultaneous Launches

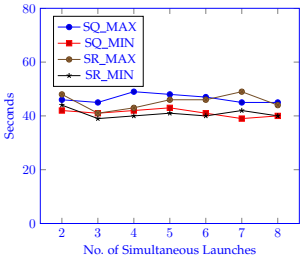


Fig. 25: Max and Min VM Booting Time under Different Simultaneous Launches

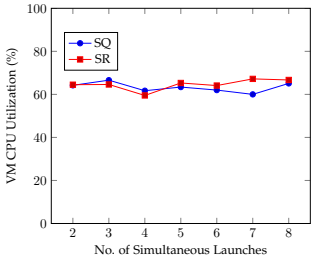


Fig. 26: VM Peak CPU Utilization under Different Simultaneous Launches

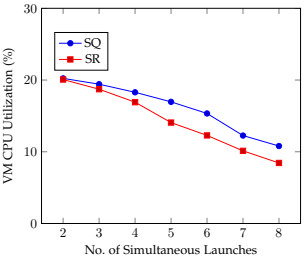


Fig. 27: VM Average Prologue CPU Utilization under Different Simultaneous Launches

cached images and cached images, respectively. It is observed that when the host disk write utilization increases, the booting time of the VMs also increases. When the host disk write utilization reaches 100%, it almost takes two times long to boot a VM when compared to the base VMs' booting time. In addition, as shown in Fig. 20 and Fig. 21, the host disk write utilization does not have much impact on the peak VM CPU utilization during booting processes.

5.6 Image Repository Impact

Under the FermiCloud architecture, as shown in Fig. 3, all the machines are connected to the SAN. Hence, when large number of VMs are launched simultaneously, all the VM images are read and copied from the SAN at the same time which may lead to significant increase of the VM's launching overheads. We set up another set of experiments to evaluate the impact of sudden large number of simultaneous launches on the VM launching overhead. Since for cached images, the images are directly read from the host machine's memory, hence, the SAN activities does not affect the VM launching process for cached image. We therefore, only evaluate uncached images. In particular, we launch a VM on a host, and simultaneously launch more VMs on other different hosts.

Fig. 22 shows the average launching times and maximum and minimum launching times for the SQ_U and SR_U test cases. X-axis presents the number of simultaneous launches. It is clear that when the number of simultaneous launches increases, the VM launching times also increase. As the number of simultaneous launches only affect the image read speed from the SAN, once the image is copied to the local host machine, the booting process becomes the same as the base VM booting process. Fig. 24 and 25 show the average VM booting time and maximum and minimum VM booting time, respectively. Fig. 26 depicts the average peak VM CPU utilization during the VM booting process. It is clear that the VMs' average booting times and average peak VM CPU utilization during the booting process are not influenced much by the number of simultaneous launches and aligned well with the base VMs' booting patterns as stated in Table 1.

As shown in Fig. 23, the number of simultaneous launches has significant impact on the image transmission process. When the number of simultaneous launches increases, the image transfer time variation also increases. This is because when the number of simultaneous launches increases, the number of requests for read bandwidth of

the SAN also increases. As a result, the read speed for each request is decreased. Fig. 27 shows when the read speed decreases, the average VM CPU utilization during the prologue process also decreases linearly. As mentioned before, the local disk read bandwidth is 500MB/s. When the read bandwidth is fully utilized, the host CPU utilization consumption is 70% on a single core. When eight VMs are being launched simultaneously, the measured average read speed on the SAN is 62MB/s for the tested VM. Hence, the theoretical VM CPU utilization during the prologue process is $62/(500/0.7) = 8.6\%$ which is consistent with the measured average value(8.4%).

5.7 Network Impact

Because the FermiCloud is using SAN as its storage, all the image transmission processes are treated as local disk activities. Hence, the network activities will not have impact on the VMs' launching process. However, the SAN architecture itself is a special network where the network bandwidth is much larger than the disk write/read speed and the network bandwidth is fully dedicated to its connected machines. In the environment where the machines are connected by the high speed Ethernet, the actual network bandwidth (at GB level) is still much larger than the disk write/read speed (at hundreds MB level). We also conducted the same set of experiments that we have done on the SAN architecture FermiCloud is built upon on the Ethernet environment [21]. The experimental results are the same as the observations we have on the SAN architecture. The disk write/read speed is also the dominant factor that significantly impact the VMs' launching overhead.

5.8 Discussion

From these experiments, we can conclude the following:

- *VM launching overhead mainly contains two parts: prologue (image copying/transferring) overhead and booting overhead;*
- *booting overhead is relatively steady, i.e., has less variations when it is compared to the prologue overhead;*
- *prologue overhead, on the other hand, has significant variations when the disk read speed on image repository varies; and*
- *disk write utilization has significant impact on both prologue overhead and booting overhead.*

As [5] states that different VM and cloud management tools have significant performance differences on VM process. In the experiments, we are trying to minimize the impact of OpenNebula by discounting its default scheduler overhead from VM launching process. We do believe that different VM management tools may impact VM performance on resource consumption. Using different VM management tool, for instance, Xen may lead different peak CPU utilization during the VM booting process, but we believe that the patterns of the image transferring process and VM booting process are the same regardless of the VM and cloud management tools.

Another major factor that impacts the VM launching process is the cloud infrastructure. In our earlier work, we have performed the same set of experiments on a different cloud infrastructure where host machines and image repository are connected by Ethernet and regular NFS file system [21]. The patterns we have observed from VM launching process are the same on both cloud infrastructures. Hence, we believe that the patterns of VM launching process are generally the same in most of the private cloud systems. One of our future work is to verify the hypothesis.

In the next section, we present a reference model for the VM launching overhead based on the data obtained.

6 VM Launching Overhead Reference Model

Before we present the reference model for the VM launching overhead in private cloud, we first introduce notations to be used in defining the reference model.

A VM in a private cloud is defined as $v_i = (f_i, t_i, H_i)$, where f_i is the image size of the VM v_i, t_i is the VM start time and H_i is the host machine that the VM is to be deployed on. For each host H_i, we use $V_{H_i} = \{v_1, v_2, \ldots, v_n\}$ to denote the set of VMs deployed on the host H_i. In the set V_{H_i}, virtual machines are sorted according to their start time in non-decreasing order. For each host H_i, p_i and m_i are the number of cores and the total memory owned by the host H_i, respectively. The $B^w_{H_i}$, $B^r_{H_i}$ and $B^c_{H_i}$ denote host machine disk write bandwidth, disk read bandwidth, and cache bandwidth, respectively; $S_w(H_i, t)$, $S_r(H_i, t)$ and $S_c(H_i, t)$ denote the file write speed, file read speed, and file cache speed on host H_i at time t, respectively.

The proposed reference model contains three different overheads: CPU utilization overhead, disk write utilization overhead, and timing overhead which is the time needed for launching a VM untill it is ready to execute tasks. We first model the CPU utilization and disk write utilization that a single VM consumes on the host machine during the launching process. Then we model the host machine's entire system CPU utilization and disk write utilization. As discussed in section 5, the complete VM launching process mainly consists of two parts: prologue and boot process. We discuss the reference models for these two steps below.

6.1 Prologue Overhead Model

The prologue overhead we model here also contains three different parts: CPU utilization overhead, disk write utilization overhead, and timing overhead which is the time needed for transferring an image from image repository to the host machine. Since the prologue overhead for uncached image and cached image are different, we have separate models for the uncached and the cached images, respectively.

6.1.1 Uncached Image

The image transferring time for VM $v_i = \{f_i, t_i, H_i\}$ with uncached image is defined below:

$$Trans_i = \frac{f_i}{S_r(H_i, t_i)} \quad (1)$$

From the experiments we know that if local disk read bandwidth is fully utilized by a process, the process also utilizes 70% of one physical core of the host machine. For

different systems, the ratio may vary. We denote such ratio as β. For a given system, the β is a constant. We first define the base VM CPU utilization of prologue as follows:

$$U_{b\_pro}(i,t) = \frac{1}{1+e^{-0.5(Trans_i+t_i)(t-t_i)}} - \frac{1}{1+e^{-0.5(Trans_i+t_i)(t-(Trans_i+t_i))}} \quad (2)$$

From the observation of image transferring process in previous section we know that when the image transferring process starts, it consumes all the I/O utilization in a very short period and occupied the I/O resources until it finishes transferring, then it releases the I/O resource. Equation (2) is given to match such utilization variations. The VM CPU utilization of transferring an image for VM v_i is modeled as:

$$U_{tr}(i,t) = \begin{cases} \frac{S_r(H_i,t_i)}{B^r_{H_i}/\beta} * U_{tr\_base}(i,t) & t_i \le t \le t_i + Trans_i \\ 0 & otherwise \end{cases} \quad (3)$$

The disk write utilization consumed by a VM's prologue process is more complicated. As the write process is not synchronized with the prologue process, we first need to determine the start time of the disk write process. As discussed above, by Linux default setting, a file is written into disk at the time when the dirty page reaches 10% of the memory or 30 seconds whichever happens first. Hence, we define the start time of write process for VM v_i as:

$$st^{UC}_w(v_i) = t_i + \min\{30, \frac{\min\{f_i, m_i/10\}}{S_r(H_i,t_i)}\} \quad (4)$$

We then need to define the time points that the VM launching process actually writes disk. As discussed above, if the disk write speed is larger than the read speed, the write process sleeps for 5 seconds after writing all the content in the dirty page. We calculate the first time duration that the process writes file into the disk.

$$fst^{UC}_w(v_i) = \min\{\frac{f_i}{S_w(H_i,t_i)}, \min\{30, \frac{\min\{f_i, m_i/10\}}{S_r(H_i,t_i)}\} \cdot \frac{S_r(H_i,t_i)}{S_w(H_i,t_i) - S_r(H_i,t_i)}\} \quad (5)$$

After the first write duration, the remaining size of the file is $rf_i = f_i - fst^{UC}_w(v_i) * S_w(H_i,t_i)$. Hence, the total time of writing the remaining file into the disk is $rf_i/S_w(H_i,t_i)$. The write time after each 5 second sleep is $wt_{sleep} = 5 * S_r(H_i,t_i)/(S_w(H_i,t_i) - S_r(H_i,t_i))$. Then the total number of sleeps is $N_{sleep} = \lceil rf_i/S_w(H_i,t_i)/wt_{sleep} \rceil$. Hence, we can obtain the time points that the entire write process finishes as below:

$$lt^{UC}_w(v_i) = st^{UC}_w(v_i) + fst^{UC}_w(v_i) + rf_i/S_w(H_i,t_i) + N_{sleep} * 5 \quad (6)$$

The time point set that the write process actual writes file into disk is defined as:

$$\begin{aligned} \mathcal{T} = \{&[st^{UC}_w(v_i), st^{UC}_w(v_i) + fst^{UC}_w(v_i)] \cup \\ &[st^{UC}_w(v_i) + fst^{UC}_w(v_i) + 1 * 5, \\ &st^{UC}_w(v_i) + fst^{UC}_w(v_i) + 1 * 5 + wt_{sleep}] \cup \ldots \cup \\ &[st^{UC}_w(v_i) + fst^{UC}_w(v_i) + (N_{sleep} - 1) * 5, \\ &st^{UC}_w(v_i) + fst^{UC}_w(v_i) + (N_{sleep} - 1) * 5 + wt_{sleep}] \cup \\ &[st^{UC}_w(v_i) + fst^{UC}_w(v_i) + N_{sleep} * 5, \\ &st^{UC}_w(v_i) + fst^{UC}_w(v_i) + N_{sleep} * 5 \\ &\qquad + (rf_i/S_w(h_i,t_i)) mod wt_{sleep}]\} \end{aligned} \quad (7)$$

At last, we define the disk write utilization consumed by the VM prologue process and the host CPU utilization it consumes. We first define the base host disk write utilization as:

$$IO^{UC}_{w\_base}(i,t) = \frac{1}{1+e^{-0.5(lt^{UC}_w)(t-st^{UC}_w(v_i))}} - \frac{1}{1+e^{-0.5(lt^{UC}_w)(t-lt^{UC}_w(v_i))}} \quad (8)$$

Then the host disk write utilization for the prologue process is defined as:

$$IO^{UC}_w(i,t) = \begin{cases} \frac{S_w(H_i,t_i)}{B^w_{H_i}} * IO^{UC}_{w\_base}(i,t) & t \in \mathcal{T} \\ 0 & otherwise \end{cases} \quad (9)$$

The host CPU utilization consumption of the write process is:

$$U^{UC}_w(i,t) = \frac{1}{p_i} IO^{UC}_w(i,t) \quad (10)$$

6.1.2 Cached Image

The prologue time and VM CPU utilization during the prologue process can be calculated using equation (1) and (3) by replacing the $S_r(h_i,t_i)$ with $S_c(h_i,t_i)$, respectively. For the host disk write utilization during the prologue process, the model is much simpler compared with the model for the uncached images.

The start time point $st_w(v_i)$ of the write process also can be calculated using equation (4) by replacing $S_r(h_i,t_i)$ with $S_c(h_i,t_i)$. The finish time point of the write process is:

$$lt_w(v_i) = st_w(v_i) + \frac{f_i}{S_c(H_i,t_i)} \quad (11)$$

Hence, the base host disk write utilization for cached images is defined as:

$$IO_{w\_base}(i,t) = \frac{1}{1+e^{-0.5(lt_w)(t-st_w(v_i))}} - \frac{1}{1+e^{-0.5(lt_w)(t-lt_w(v_i))}} \quad (12)$$

Then the host disk write utilization for the prologue process is:

$$IO^C_w(i,t) = \begin{cases} \frac{S_c(H_i,t_i)}{B^c_{H_i}} * IO_{w\_base}(i,t) & st_w(v_i) \le t \le lt_w(v_i) \\ 0 & otherwise \end{cases} \quad (13)$$

The host CPU utilization consumption of the write process is:

$$U_w^C(i,t) = \frac{1}{p_i} IO_w^C(i,t) \quad (14)$$

In general, the host disk write utilization during the VM prologue process is defined as:

$$IO_w(i,t) = \begin{cases} IO_w^{UC}(i,t) & \textit{uncached image} \\ IO_w^C(i,t) & \textit{cached image} \end{cases} \quad (15)$$

The host CPU utilization consumption of the write process is:

$$U_w(i,t) = \begin{cases} U_w^{UC}(i,t) & uncachedimage \\ U_w^C(i,t) & cachedimage \end{cases} \quad (16)$$

Note that the model for uncached images only validates for the scenario when $S_w(H_i,t_i) > S_r(H_i,t_i)$ and images are not cached. When $S_w(H_i,t_i) \leq S_r(H_i,t_i)$, we need to use the cached image model by replacing $S_c(H_i,t_i)$ with $S_r(H_i,t_i)$.

6.2 Booting Overhead Model

The VM booting overhead also refers to the timing overhead and CPU utilization overhead. As once the image is copied to a host, it will not consume any disk write utilization for the booting process. We consider that there is no disk write overhead for the virtual machine booting process. The experiments also indicate that the host disk write utilization impacts the booting overhead. Hence, we model the CPU utilization overhead for the VM v_i's booting process as follows:

$$U_b(i,t) \quad (17)$$
$$= \begin{cases} a*(t - Trans_i - t_i) & , t < Trans_i + t_i + \frac{f(v_i)}{a} \\ f(v_i)\frac{1}{m} e^{-t'\gamma(\alpha(1+IO_s(H_i,t-1))+\frac{\lambda}{t'+\lambda})} & , otherwise \end{cases}$$

where $f(v_i)$ is the function related to the virtual CPU cores that v_i has and it is used to control the peak VM CPU utilization during the booting process. In equation(17), a, γ, α and λ are constants, and γ and λ dominate the function decay rate while α determines the minimum decay rate, m is the number of cores on the host machine and $IO_s(H_i,t-1)$ represents the system's disk write utilization at time $t-1$. $t' = t - Trans_i - t_i - f(v_i)/a$. We will formally define the system disk IO utilization in section 6.5.

In OpenNebula, VMs are not immediately ready for use until all the necessary services, such as *SSHD*, are started. As there is no accurate way to tell the actual time when a virtual machine is booted and ready for use unless entering into a running virtual machine and check the log, therefore, we base our estimation of the time points on the variation of the VM's CPU utilization consumption. If the VM's CPU utilization consumption remain stable, then we consider the VM is booted and ready for use. We define the time point $t_b(i)$ at which a VM v_i is ready to use as:

$$t_b(i) = \max\{t || U_b'(i,t)| \leq \epsilon\} \quad (18)$$

where $U_b'(i,t)$ is the first derivative of $U_b(i,t)$ and ϵ is the threshold to determine whether the virtual machine's CPU utilization consumption become stable. Then, we can calculate the VM booting time is as $(t_b(i) - Trans_i)$.

6.3 Virtual Machine Launching Overhead Model

We have formally modeled image transfer overhead and virtual machine booting overhead. Combining the two components together, we derive VM launching overhead functions. In particular, combining equation (3) and equation (17), the VM v_i's launching CPU utilization function is modeled as:

$$U(i,t) = \begin{cases} U_{tr}(i,t) & t \leq t_{tran} \\ U_b(i,t) & t > t_{tran} \end{cases} \quad (19)$$

Since IO utilization consumed by the VM booting process is negligible, the IO utilization function for VM v_i's launching process is the same as equation (15).

The total time needed for launching a VM v_i then can be calculated as image copying time plus VM booting time. It is formally defined as follow:

$$t_{overhead}(i) = t_b(i) - t_i \quad (20)$$

6.4 Virtual Machine Utilization Consumption Model

The complete VM utilization functions consist of the VM launching overhead utilization functions and the utilization functions after workloads are deployed on the virtual machine. We assume at time $t' \geq t_b(i)$, the VM v_i starts executing tasks; and the CPU and disk IO utilization consumption function of v_i at t' are $U_e(t)$ and $IO_e(t)$, respectively. Then the VM CPU utilization consumption model is defined below:

$$U_c(i,t) = \begin{cases} U_{tr}(i,t) & t \leq t_{tran} \\ U_b(i,t) & t > t_{tran} \\ U_e(i,t) & t \geq t' \end{cases} \quad (21)$$

The VM IO utilization consumption model is defined as:

$$IO_c(i,t) = \begin{cases} IO_w(i,t) & t_i \leq t \leq t_b(i) \\ IO_e(i,t) & otherwise \end{cases} \quad (22)$$

6.5 System Utilization Model

We assume that host machines only run VMs and all other critical system services consume a small portion of the system CPU and IO utilization. Then we can calculate the system CPU and disk IO utilization as the summation of the VMs' CPU and IO utilization consumptions. The system CPU utilization of host h_i is modeled below:

$$U_s(H_i,t) = \max\{1, \sum_{j=1}^{|V_{H_i}|}\{U_c(j,t)\} + \sum_{j=1}^{|V_{H_i}|}\{U_w(j,t)\}\} \quad (23)$$

The system IO utilization of host h_i can be modeled as:

$$IO_s(H_i,t) = \max\{1, \sum_{j=1}^{|V_{H_i}|}\{IO_c(j,t)\}\} \quad (24)$$

Intuitively, the mathematical equations used to model the VM launching process in this section are obtained by finding the best match equations to the plotted data shown in Section 5. There may exist other equations that can also represent the variations of the real data. To balance the trade offs between accuracy and complexity, we choose the equations that we believe give fair accuracy within a short computational time period. In next section, we use data obtained from real operation cloud to verity the accuracy of the developed reference model.

7 Evaluation

We build the reference model for virtual machine launching overhead based on a large amount of data obtained from a real production system. However, we cannot guarantee the accuracy of the model unless we compare the calculated data using the model we built with the real system data and prove the accuracy of the model. Since some of the parameters we use for modeling are system dependent, we focus the evaluation on the same given system, i.e., FermiCloud. We first use base VM launching overhead values shown in Section 5 to determine all the parameters. Once the parameters are determined, they are fixed for all the evaluation experiments. To evaluate the performance of the developed model, we first launch VMs on FermiCloud under different system loads. Then we use the developed reference model to simulate the launch process use the same VM release pattern.

We use mean square weighted deviation to evaluate the accuracy of our developed model from four aspects, i.e., VM CPU Utilization, host CPU utilization, host I/O utilization and VM launching time. We denote N as the total number of sampling points. The mean square weighted deviation is defined as follows:

$$MSWD = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{n}(U_s(i)-U_r(i,j))^2}{\sigma^2} \qquad (25)$$

where n is the number of the repetition of one experiment. $U_r(i,j)$ is the real data from the i^{th} sampling point and j^{th} repetition. U_s represents the calculated data at i^{th} sampling point. σ is the standard deviation. To check the real time point for the VM that is ready for use, we use the VM system log to check the starting time point of the *SSHD* service.

We first compare the base overhead obtained by calculating the value based on the model proposed in section 6, and the real data obtained on FermiCloud.

Fig. 28(a), Fig. 28(b), Fig. 28(c), Fig. 28(d), Fig. 29(a), Fig. 29(b), Fig. 29(c), and Fig. 29(d), draws the host CPU utilization, host disk write utilization and VM CPU utilization for SQ_U SQ_C LQ_U LQ_C SR_U SR_C LR_U LR_C test cases using the developed VM launching overhead model, respectively. Compare the graph with the utilization variations shown in Fig. 4, Fig. 5, Fig. 6, and Fig. 7 obtained from the real operation data from FermiCloud. The calculated data using our developed model is very close to the real data.

Table 2 gives a more detailed comparison between the real data and calculated data. From the table, we can observe that the maximum mean square weighted deviation for the VM CPU utilization of base test cases is 4.55 and the minimum mean square weighted deviation for the base test cases is 0.41. As the real data for the base cases shown in Fig. 4(a) to Fig. 7(b), the VM CPU utilization during the booting process drops for a small duration after it reaches the peak utilization, then immediately rises a little bit and finally decreases continuously. While in our model, the VM CPU utilization for the booting process keeps decreasing after reaches its peak utilization. Hence the range of the mean square weighted deviation for the VM CPU utilization for the base cases is from 0.41 to 4.55. The average mean square weighted deviation for VM CPU utilization of all base cases is 2.67. While the mean square weighted deviation for host disk write utilization for base cases only varies from 0.34 to 2.12. And the average mean square weighted deviation for host disk write utilization for all base cases is 1.54. As a result, the average mean square weighted deviation of host CPU utilization for all base cases is 2.19. The range of the mean square weighted deviation for the predicted VM launching time for base cases is from 2.35 to 4.21. The average mean square weighted deviation for VM launching time predicted by our model for all base cases is 2.93.

	VM CPU Util.	Host CPU U.	Host IO U.	Time
Base SQ_U	2.79	2.32	1.61	2.35
Base SQ_C	1.31	2.02	1.61	3.16
Base LQ_U	4.28	0.71	1.60	4.21
Base LQ_C	1.88	2.53	1.86	2.87
Base SR_U	1.62	2.13	2.03	3.18
Base SR_C	4.53	3.55	2.12	2.58
Base LR_U	0.41	1.12	1.15	2.41
Base LR_C	4.55	3.21	0.34	2.85
Sim. Lau.	2.39	1.98	1.38	2.43
Rand. Lau.	1.78	1.83	1.40	2.27
Overall	2.55	2.14	1.51	2.82

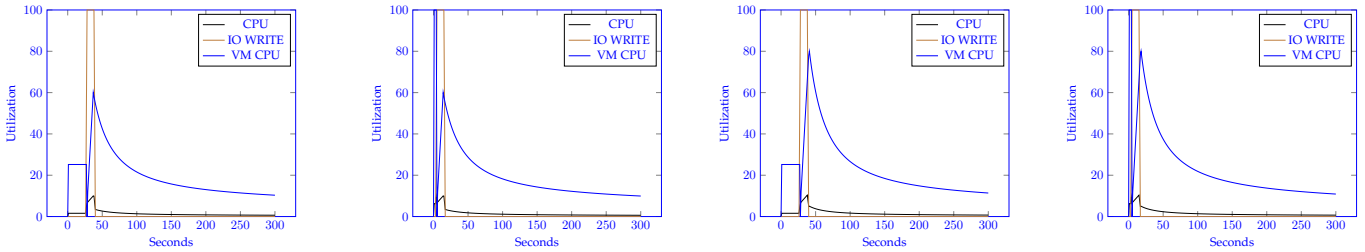
TABLE 2: Mean Square Weighted Deviation for Calculated Overheads in VM CPU Util., Host CPU Util., Host IO Util., and Time

We further evaluate when more than one VMs are launched on the same host machine simultaneously. The number of simultaneous launch increases from 2 to 4. The VMs that to be launched are arbitrary selected from our four different test cases. The obtained data is given in Table 2. The mean square weighted deviation for calculated overheads in VM CPU utilization, host CPU utilization, host disk write utilization and VM launching time are 2.39, 1.98, 1.38 and 2.43, respectively. The performance of our developed model remain the same level compared to the base test cases (average mean square deviation for VM CPU utilization, host CPU utilization, host disk write utilization and VM launching time are 2.67, 2.19, 1.54 and 2.93, respectively).

For the last set of evaluations, we launch VMs under a random release pattern. Differ from previous evaluations, as soon as a VM is launched, it immediately executes an application deployed on it. Hence, the host machine has different CPU and IO utilization at different time instances when a new VM is released. We use the reference model to simulate the VM launching process in a real cloud environment using the same release pattern. From the Table 2, it is clear that the developed reference model accurately reflects the VM launching overhead, the mean square weighted deviation of calculated data is less than 2.5 from all aspects.

Overall, for all our test cases, the average mean square weighted deviation for VM CPU utilization is 2.55, the average mean square weighted deviation for host disk write utilization is 1.51, the average mean square weighted deviation for host CPU utilization is 2.14, and the average mean square weighted deviation for launch time is 2.82. With an average mean square weighted deviation less than 3 from all four aspects, we believe that our developed model can accurately reflect the VM launching process.

8 Conclusion



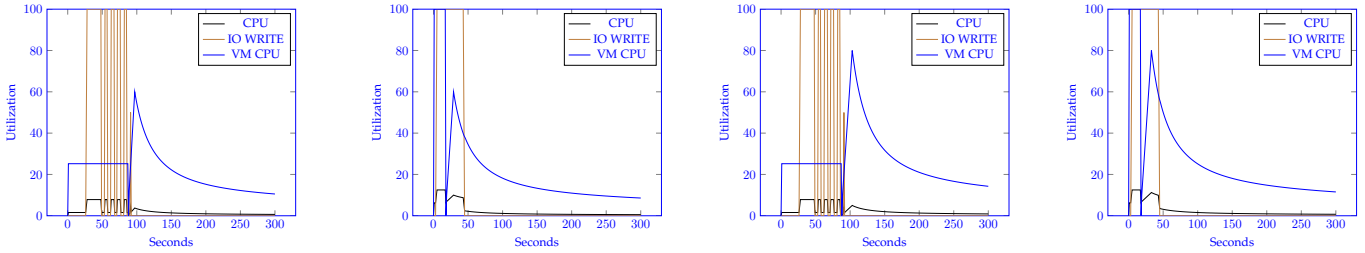
(a) VM launching overhead with SQ_U using proposed model

(b) VM Launching Overhead with SQ_C using Developed Model

(c) VM Launching Overhead with LQ_U using Developed Model

(d) VM Launching Overhead with LQ_C using Developed Model

Fig. 28: VM Launching Overhead with QCOW2 Image using Developed Model



(a) VM Launching Overhead with SR_U using Developed Model

(b) VM Launching Overhead with SR_C using Developed Model

(c) VM Launching Overhead with LR_U using Developed Model

(d) VM Launching Overhead with LR_C using Developed Model

Fig. 29: VM Launching Overhead with Raw Image using Developed Model

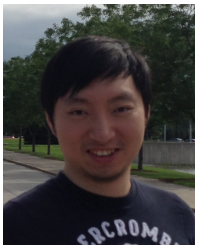
One of the main challenges in developing the cloud bursting module is to decide *when* and *where* to launch a VM so that all resources are most effectively utilized and the system performance is optimized. We have found that the VM launching overhead has a large variation under different system states. The CPU and I/O utilizations caused by VM launching process can have significant impact on cloud bursting strategies. Hence, being able to model accurately the dependency between VM launching overhead and system resource utilization is critical in deciding *when* and *where* a VM should be launched. This paper has studied the VM launching overhead patterns based on data obtained on FermiCloud and presented a VM launching overhead reference model to represent such overhead. The evaluation shows that our proposed model can accurately predict the VM launching overhead within a mean square weighted deviation less than 3 from all four aspects, i.e. VM CPU utilization, system CPU utilization, system I/O utilization and VM launching time.

The model developed in this paper is based on SAN-based cloud infrastructures with OpenNebula and KVM as its cloud and VM management tool, respectively. We do believe that the patterns we modeled for VM launching process is applicable to other private cloud in general. It is our future work to verify our hypothesis that the model does fit different virtualization techniques, i.e. fully virtualization and hardware-assisted virtualization; different cloud management tools, i.e., OpenStack etc.; different VM management tools, i.e., Xen; and different cloud infrastructures. In addition, following the motivation of developing the reference model, our immediate work is to integrate the reference model into the cloud bursting decision algorithms.

REFERENCES

[1] Feature - clouds make way for STAR to shine. http://www.isgtw.org/feature/isgtw-feature-clouds-make-way-star-shine.
[2] https://www.scientificlinux.org/.
[3] Opennebula. http://opennebula.org.
[4] Opennebula managing virtual machines. http://opennebula.org/documentation:archives:rel3.0:vm_guide_2.
[5] P. Armstrong, A. Agarwal, A. Bishop, A. Charbonneau, R. Desmarais, K. Fransham, N. Hill, I. Gable, S. Gaudet, S. Goliath, et al. Cloud scheduler: a resource manager for distributed compute clouds. *arXiv preprint arXiv:1007.0050*, 2010.
[6] A. Beloglazov and R. Buyya. Energy efficient allocation of virtual machines in cloud data centers. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 577–578. IEEE, 2010.
[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.
[8] R. N. Calheiros, R. Ranjan, and R. Buyya. Virtual machine provisioning based on analytical performance and qos in cloud computing environments. In *Parallel Processing (ICPP), 2011 International Conference on*, pages 295–304. IEEE, 2011.
[9] N. Huber, M. von Quast, M. Hauck, and S. Kounev. Evaluating and modeling virtualization performance overhead for cloud environments. In *CLOSER*, pages 563–573, 2011.
[10] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Scientific workflow applications on amazon ec2. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 59–66. IEEE, 2009.

[11] H. A. Lagar-Cavilla, J. A. Whitney, R. Bryant, P. Patchin, M. Brudno, E. de Lara, S. M. Rumble, M. Satyanarayanan, and A. Scannell. Snowflock: Virtual machine cloning as a first-class cloud primitive. *ACM Transactions on Computer Systems (TOCS)*, 29(1):2, 2011.
[12] J. Lauret, M. Walker, S. Goasguen, and L. Hajdu. From grid to cloud, the star experience, 2010.
[13] M. Mao and M. Humphrey. A performance study on the vm startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE, 2012.
[14] Y. Z. Mengxia Zhu, Qishi Wu. A cost-effective scheduling algorithm for scientific workflows in cloud. *Proceedings of 31st IEEE International Performance Computing and Communications Conference*, 2012.
[15] R. Moreno-Vozmediano, R. Montero, and I. Llorente. Iaas cloud architecture: from virtualized data centers to federated cloud infrastructures. 2012.
[16] S.-Y. Noh, S. C. Timm, and H. Jang. vcluster: a framework for auto scalable virtual cluster system in heterogeneous clouds. *Cluster Computing*, pages 1–9, 2013.
[17] J. Qiu, J. Ekanayake, T. Gunarathne, J. Y. Choi, S.-H. Bae, H. Li, B. Zhang, T.-L. Wu, Y. Ruan, S. Ekanayake, et al. Hybrid cloud and cluster computing paradigms for life science applications. *BMC bioinformatics*, 11(Suppl 12):S3, 2010.
[18] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya. Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pages 279–287. IEEE, 2011.
[19] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman. Experiences using cloud computing for a scientific workflow application. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, pages 15–24. ACM, 2011.
[20] H. Wu, S. Ren, G. Garzoglio, S. Timm, G. Bernabeu, H. W. Kim, K. Chadwick, H.-J. Jang, and S.-Y. Noh. Automatic cloud bursting under fermicloud. *Workshop on Cloud Services and Systems*, 2013.
[21] H. Wu, S. Ren, G. Garzoglio, S. Timm, G. Bernabeu, and S.-Y. Noh. A reference model for virtual machine launching overhead. In *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014)*, 2014.



Hao Wu is now a Ph.D candidate in Computer Science Department at Illinois Institute of Technology. He received B.E in Information Security from Sichuan University, Chengdu, China, 2007. He received M.S. in Computer Science from University of Bridgeport, Bridgeport, CT, 2009. His current research interests mainly focus on cloud computing, real-time distributed open systems, Cyber-Physical System, parallel and distributed systems, and real-time applications.

Dr. Shangping Ren is an associate professor in Computer Science Department at the Illinois Institute of Technology. She earned her Ph.D from UIUC in 1997. Before she joined IIT in 2003, she worked in software and telecommunication companies as software engineer and then lead software engineer. Her current research interests include coordination models for real-time distributed open systems, real-time, fault-tolerant and adaptive systems, Cyber-Physical System, parallel and distributed systems, cloud computing, and application-aware many-core virtualization for embedded and real-time applications.

Dr. Gabriele Garzoglio is head of the Grid and Cloud Services Department of the Scientific Computing Division at Fermilab and he is deeply involved in the project management of the Open Science Grid. He oversees the operations of the Grid services at Fermilab and sponsors the Cloud program in the division. Gabriele Garzoglio has a Laura degree in Physics from University of Genova, Italy, and a PhD in Computer Science from DePaul University, Chicago.

Dr. Steven Timm is the associate head for Cloud Computing of the Grid and Cloud Services Department at Fermilab. He has led the FermiCloud project since its inception in early 2010, and been a member of the Fermilab staff since 2000. In this role he coordinates working with visiting students and guest researchers from other laboratories doing research and development on innovative techniques in distributed computing and cloud computing. He completed his PhD. Studies at Carnegie Mellon University in 1995.

Gerard Bernabeu is an IT engineer and researcher of the Grid and Cloud Services Department of the Scientific Computing Division at Fermilab. He is Linux DevOps enthusiast with hand-on experience in IP networks, storage and Cloud Computing in dynamic, collaborative research communities. He received MsC in High Performance Computing and Information Theory by the Universitat Autnoma de Barcelona (UAB).

Dr. Keith Chadwick is ITIL Avilability and Service Continuity Manager at fermilab. During 2009 - 2013, he was the head of the Grid and Cloud Services Department of the Scientific Computing Division at Fermilab. He has been a member of the Fermilab staff since 1987. Dr. Keith received B.S. and M.A. in Physics from the University of Rochester in 1978 and 1980, respectively. He received Ph.D in Physics from the University of Rochester in 1984.

Dr. Seo-Young Noh is a principal researcher in National Institute of Supercomputing and Networking at Korea Institute of Science and Technology Information and an associate professor at Korea University of Science and Technology. He is leading the development of virtual cluster system called vcluster in conjunction with KISTI-FNAL joint project. Before joining the institutes, he worked for LG Electronics in the fields of embedded database systems and Linux mobile platforms. He received his B.E and M.E in Computer Engineering from Chungbuk National University in Korea and his M.S. and Ph.D. in Computer Science from Iowa State University, respectively. His research interests are including scientific data management, cloud & scientific computing, Linux platforms, databases, and natural language processing.